KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

(Approved by AICTE & Govt of T.S. and Affiliated to JNTUH)
NARAYANAGUDA, HYDERABAD – 500 029

Certificate

This is to certify that following is a Bonafide Record of the practical work done by

_____bearing Roll

No._____ of _____ Branch of _____ year B.Tech

Course in the_____ Laboratory during the Academic year

_____ & _____ under our supervision.

Number of experiments conducted: _____

Signature of Staff Member Incharge

Signature of Head of the Dept.

Signature of Internal Examiner

Signature of External Examiner

INDEX

PARTICULARS OF THE EXPERIMENT PERFORMED

SNO	Name of the Experiment	Page No	Date of Experiment	Signature of faculty

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

Daily Laboratory Assessment Sheet

Name of the Lab: **Name of the Student:**

Class: **Roll No:**

S.No	Name of the Experiment	Date	Record Marks	Viva Voice Marks	Total Marks	Signature of Faculty

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
(Approved by AICTE & Govt of T.S and Affiliated to JNTUH)
3-5-1026, Narayanaguda, Hyderabad-29. Ph: 040-23261407

Department of Computer Science and Engineering

Vision & Mission of the Department

Vision of the Department

To be among the region's premier teaching and research Computer Science and Engineering departments producing globally competent and socially responsible graduates in the most conducive academic environment.

Mission of the Department

- To provide faculty with state of the art facilities for continuous professional development and research, both in foundational aspects and of relevance to emerging computing trends.
- To impart skills that transform students to develop technical solutions for societal needs and inculcate entrepreneurial talents.
- To inculcate an ability in students to pursue the advancement of knowledge in various specializations of Computer Science and Engineering and make them industry-ready.
- To engage in collaborative research with academia and industry and generate adequate resources for research activities for seamless transfer of knowledge resulting in sponsored projects and consultancy.
- To cultivate responsibility through sharing of knowledge and innovative computing solutions that benefit the society-at-large.
- To collaborate with academia, industry and community to set high standards in academic excellence and in fulfilling societal responsibilities.

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY

(Approved by AICTE & Govt of T.S and Affiliated to JNTUH)
3-5-1026, Narayanaguda, Hyderabad-29. Ph: 040-23261407

Department of Computer Science and Engineering

PROGRAM OUTCOMES (POs)

1. **Engineering Knowledge:** Apply knowledge of mathematics and science, with fundamentals of Computer Science & Engineering to be able to solve complex engineering problems related to CSE.
2. **Problem Analysis:** Identify, Formulate, review research literature and analyze complex engineering problems related to CSE and reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences
3. **Design/Development of solutions:** Design solutions for complex engineering problems related to CSE and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety and the cultural societal and environmental considerations
4. **Conduct Investigations of Complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern Tool Usage:** Create, Select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to computer science related complex engineering activities with an understanding of the limitations
6. **The Engineer and Society:** Apply Reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the CSE professional engineering practice
7. **Environment and Sustainability:** Understand the impact of the CSE professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development

8. **Ethics:** Apply Ethical Principles and commit to professional ethics and responsibilities and norms of the engineering practice
9. **Individual and Team Work:** Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary Settings
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large such as able to comprehend and with write effective reports and design documentation, make effective presentations and give and receive clear instructions.
11. **Project Management and Finance:** Demonstrate knowledge and understanding of the engineering management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi-disciplinary environments
12. **Life-Long Learning:** Recognize the need for and have the preparation and ability to engage in independent and life-long learning the broadest context of technological change

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
(Approved by AICTE & Govt of T.S and Affiliated to JNTUH)
3-5-1026, Narayanaguda, Hyderabad-29. Ph: 040-23261407

Department of Computer Science and Engineering

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: An ability to analyze the common business functions to design and develop appropriate Computer Science solutions for social upliftments.

PSO2: Shall have expertise on the evolving technologies like Python, Machine Learning, Deep Learning, Internet of Things (IOT), Data Science, Full stack development, Social Networks, Cyber Security, Big Data, Mobile Apps, CRM, ERP etc.

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
(Approved by AICTE & Govt of T.S and Affiliated to JNTUH)
3-5-1026, Narayanaguda, Hyderabad-29. Ph: 040-23261407

Department of Computer Science and Engineering

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO 1: Graduates will endeavour to excel in their chosen careers as professionals, researchers and entrepreneurs on a global platform.

PEO 2: Graduates will demonstrate the ability to solve challenges in the fields of Engineering and Technology simultaneously catering to societal needs.

PEO 3: Graduates will strive to improve their learning curve by practising Continuing Professional Development (CPD)

PEO 4: Graduates will, at all times, adopt a professional demeanor by communicating effectively, working collaboratively, and maintaining the ethics & core values as befitting their education in interdisciplinary and emerging fields.

KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
(Approved by AICTE & Govt of T.S and Affiliated to JNTUH)
3-5-1026, Narayanaguda, Hyderabad-29. Ph: 040-23261407

Department of Computer Science & Engineering

Course Outcomes and CO-PO Mapping

Academic year: 2020-21 **Year&Sem:- III - II**

Subject Name: COMPILER DESIGN **Branch & Section: CSE –**

Course Outcomes:

1.Demonstrate the ability to design a compiler given a set of language features.

2.Demonstrate the knowledge of patterns, tokens & regular expressions for
lexical analysis.

3. Acquire skills in using lex tool & yacc tool for developing a scanner and parser.

4. Design and implement LL and LR parsers

5. Design algorithms to do code optimization in order to improve the performance of
a program in terms of space and time complexity.

6. Design algorithms to generate machine code

CO-PO-PSO Matrix:

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO10	PO11	PO12	PSO 1	PSO 2
CO 1	3	1		2									2	
CO 2	2	3	3	2									1	
CO 3		2	3										2	
CO 4	2			2								2		
CO 5	3	2			2								1	

Signature of Faculty **Signature**
of Subject-Coordinator

CO – PO MAPPING JUSTIFICATION

MAPPING	LOW (1) MEDIUM (2) HIGH (3)	JUSTIFICATION
CO 1- PO 1	3	Knowledge is applied to understand the machines power to find the solutions for engineering problems.
CO 1- PO 2	1	Concept of abstract machines is useful to analyze the mathematical problems.
CO 1- PO 4	2	Concepts of abstract machines are used to design the complex problems.
CO 1- PSO 1	2	Engineering knowledge used to analyze the computer science problems.
CO 2- PO 1	2	FA model applied to find the solutions for computing problems.
CO 2- PO 2	3	FA machines are useful to identify and formulate the mathematical problems.
CO 2- PO 3	3	FA model applied to design and develop solutions for complex engineering problems.
CO 2- PO 4	2	Knowledge of finite state machines applied to analyze and solve complex problems.
CO 2- PSO 1	1	Finite automata model used to develop solutions for the computer science problems.
CO 3- PO 2	2	Knowledge of grammars is used to find the solutions of the engineering science problems.

CO 3- PO 3	3	Grammars information applied to design solutions for engineering problems.
CO 3- PSO 1	2	Information of grammars applied to analyze and model the computer science problems.
CO 4- PO 1	2	Knowledge applied to find differences between decidable and un-decidable problems.
CO 4- PO 4	2	Decidable and un-decidable knowledge applied to investigate complex problems.
CO 4- PO 12	2	Decidable and un-decidable knowledge applied to get solution for each engineering problems.
CO 5- PO 1	3	Mathematical and formal methods knowledge is applied to get the solutions for engineering problems.
CO 5- PO 2	2	Concepts of formal methods are useful to analyze the mathematical problems.
CO 5- PO 5	2	Proficiency of mathematical tools and formal methods are applied to predict and model the complex engineering activities.
CO 5- PSO 1	1	Modern tools are used to get solutions for the computer science problems.

Program 1:

AIM: To Write a LEX Program to scan reserved word & Identifiers of C Language.

```
/* program name is lexp.l */
%{
/* program to recognize a c program */
int COMMENT=0;
%}
identifier [a-zA-Z][a-zA-Z0-9]*
%%
#.* { printf("\n%s is a PREPROCESSOR DIRECTIVE",yytext);}
int |
float |
char |
double |
while |
for |
do |
if |
break |
continue |
void |
switch |
case |
long |
struct |
const |
typedef |
return |
else |
goto {printf("\n\t%s is a KEYWORD",yytext);}
```

```
"/*" {COMMENT = 1;}
/*{printf("\n\n\t%s is a COMMENT\n",yytext);}*/
"*/" {COMMENT = 0;}
/* printf("\n\n\t%s is a COMMENT\n",yytext);}*/
{identifier}\( {if(!COMMENT)printf("\n\nFUNCTION\n\t%s",yytext);}
\{ {if(!COMMENT) printf("\n BLOCK BEGINS");}
\} {if(!COMMENT) printf("\n BLOCK ENDS");}
{identifier}(\[[0-9]*\])? {if(!COMMENT) printf("\n %s IDENTIFIER",yytext);}
\".*\" {if(!COMMENT) printf("\n\t%s is a STRING",yytext);}
[0-9]+ {if(!COMMENT) printf("\n\t%s is a NUMBER",yytext);}
\)(\;)? {if(!COMMENT) printf("\n\t");ECHO;printf("\n");}
\( ECHO;
= {if(!COMMENT)printf("\n\t%s is an ASSIGNMENT OPERATOR",yytext);}
\<= |
\>= |
\< |
== |
\> {if(!COMMENT) printf("\n\t%s is a RELATIONAL OPERATOR",yytext);}
%%
int main(int argc,char **argv)
{
if (argc > 1)
{
FILE *file;
file = fopen(argv[1],"r");
if(!file)
{
printf("could not open %s \n",argv[1]);
exit(0);
}
yyin = file;
```

```
}
yylex();
printf("\n\n");
return 0;
} int yywrap()
{
return 0;
}
```

Input:

```
$vi var.c
#include<stdio.h>
main()
{
int a,b;
}
$lex lex.l
$cc lex.yy.c
$./a.out var.c
```

OUTPUT:

```
#include<stdio.h> is a PREPROCESSOR DIRECTIVE
FUNCTION
main ()
BLOCK BEGINS
int is a KEYWORD
a IDENTIFIER
b IDENTIFIER
BLOCK ENDS
```

Program 2:

AIM: To Implement Predictive Parsing algorithm.

```
#include<stdio.h>
#include<conio.h>
char nt[]={'E','A','T','B','F'},ter[]={'i','+','*','(',')','$'};
char arr[20][20][20]={
                {"TA"," "," ","TA"," "," "},
                {" ","+TA"," "," ","#","#"},
                {"FB"," "," ","FB"," "," "},
                {" ","#","*FB"," ","#","#"},
                {"i"," "," ","(E)"," "," "}
        };
char ipstr[20];
char stack[40],prod[10];
int i=0,top=1,ia,ix;
void main(void )
{
void pop();
void push(char );
int resolve_nt(char );
int resolve_t(char );
void advance();
char a,x;
int len,temp,k;

stack[0]='$';
stack[1]='E';
printf("Enter the input string:\n");
printf("Enter $ as an end marker\n");
scanf("%s",ipstr);
printf("I/P String\t\tStack Contents\t\tProduction Used\n");
while(1)
{
        a=ipstr[i];
        x=stack[top];
        /*To display the input string*/
        for(k=i;ipstr[k]!='$';k++)
                printf("%c",ipstr[k]);
        printf("$\t\t");
        if(x==a)
        {
                if(x=='$')
```

```
                {
                        printf("\rinput string is accepted");
                        break;
                }
                else
                {
                        pop();
                        advance();
                }
        }
        else if(isupper(x))
        {
                ix=resolve_nt(x);
                ia=resolve_t(a);
                strcpy(prod,arr[ix][ia]);
                len=strlen(prod);
                pop();
                for(k=1;k<=len;k++)
                        push(prod[len-k]);
                if(stack[top]=='#')
                        pop();
        }
        else
        {
                printf("Error: Could not parse teh input string");
                break;
        }
        /*To display the stack contents and the production used*/
        for(k=0;k<=top;k++)
                printf("%c",stack[k]);
        printf("\t\t\t\t%s\n",prod);
}
getch();
}
void push(char t)
{
top+=1;
stack[top]=t;
}
void pop()
{
top--;
}
void advance()
{
i++;
```

```
}
int resolve_nt(char t)
{
int k,index;
for(k=0;k<5;k++)
{
        if(t==nt[k])
        {
                index=k;
                break;
        }
}
return index;
}
int resolve_t(char t)
{
int k,index;
for(k=0;k<6;k++)
{
        if(t==ter[k])
        {
                index=k;
                break;
        }
}
return index;
}
```

INPUT:

Enter a string

i+i$

OUTPUT:

```
I/P String              Stack Contents          Production Used
i+i$            $AT                             TA
i+i$            $ABF                            FB
i+i$            $ABi                            i
i+i$            $AB                             i
+i$             $A                              #
+i$             $AT+                            +TA
+i$             $AT                             +TA
i$              $ABF                            FB
i$              $ABi                            i
i$              $AB                             i
$               $A                              #
$               $                               #
input string is accepted
```

Program 3:

AIM: To write a C program to generate three address code.

```
#include<stdio.h>
#include<string.h>
#include<iostream>
void pm();
void plus();
void div();
int i,ch,j,l,addr=100;
char ex[10], exp[10] ,exp1[10],exp2[10],id1[5],op[5],id2[5];
int main()
{
while(1)
{
printf("\n1.assignment\n2.arithmetic\n3.relational\n4.Exit\nEnter the choice:");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("\nEnter the expression with assignment operator:");
scanf("%s",exp);
l=strlen(exp);
exp2[0]='\0';
i=0;
while(exp[i]!='=')
{
i++;
}
strncat(exp2,exp,i);
strrev(exp);
exp1[0]='\0';
strncat(exp1,exp,l-(i+1));
strrev(exp1);
printf("Three address code:\ntemp=%s\n%s=temp\n",exp1,exp2);
break;

case 2:
printf("\nEnter the expression with arithmetic operator:");
scanf("%s",ex);
strcpy(exp,ex);
l=strlen(exp);
exp1[0]='\0';
```

```
for(i=0;i<l;i++)
{
if(exp[i]=='+'||exp[i]=='-')
{
if(exp[i+2]=='/'||exp[i+2]=='*')
{
pm();
break;
}
else
{
plus();
break;
}
}
else if(exp[i]=='/'||exp[i]=='*')
{
div();
break;
}
}
break;

case 3:
printf("Enter the expression with relational operator");
scanf("%s%s%s",&id1,&op,&id2);
if(((strcmp(op,"<")==0)||(strcmp(op,">")==0)||(strcmp(op,"<=")==0)||(strcmp(op,">=")==0)||(str
cmp(op,"==")==0)||(strcmp(op,"!=")==0))==0)
printf("Expression is error");
else
{
printf("\n%d\tif %s%s%s goto %d",addr,id1,op,id2,addr+3);
addr++;
printf("\n%d\t T:=0",addr);
addr++;
printf("\n%d\t goto %d",addr,addr+2);
addr++;
printf("\n%d\t T:=1",addr);
}
break;
case 4:
exit(0);
}
}
}
```

```
void pm()
{
strrev(exp);
j=l-i-1;
strncat(exp1,exp,j);
strrev(exp1);
printf("Three address code:\ntemp=%s\ntemp1=%c%ctemp\n",exp1,exp[j+1],exp[j]);
}
void div()
{
strncat(exp1,exp,i+2);
printf("Three address code:\ntemp=%s\ntemp1=temp%c%c\n",exp1,exp[i+2],exp[i+3]);
}
void plus()
{
strncat(exp1,exp,i+2);
printf("Three address code:\ntemp=%s\ntemp1=temp%c%c\n",exp1,exp[i+2],exp[i+3]);
}
```

Output

```
1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:1

Enter the expression with assignment operator:a=45
Three address code:
temp=45
a=temp

1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:_
```

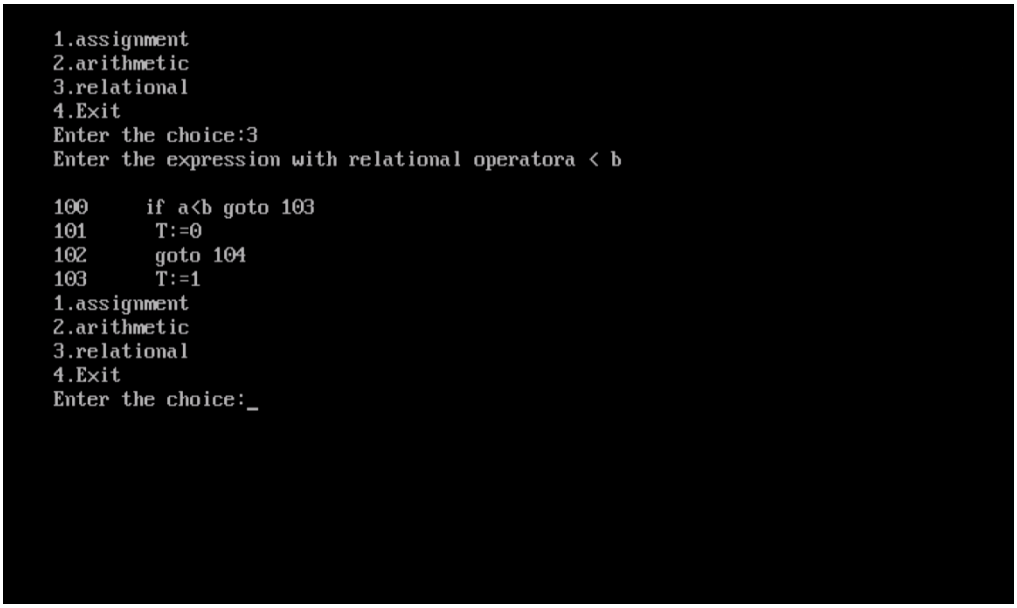
```
3.relational
4.Exit
Enter the choice:2

Enter the expression with arithmetic operator:b+c
Three address code:
temp=b+c
temp1=temp

1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:2

Enter the expression with arithmetic operator:b+c+d
Three address code:
temp=b+c
temp1=temp+d

1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:_
```


1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:3
Enter the expression with relational operatora < b
100 if a<b goto 103
101 T:=0
102 goto 104
103 T:=1
1.assignment
2.arithmetic
3.relational
4.Exit
Enter the choice:_

Program 4:

AIM: To Implement SLR(1) Parsing algorithm

```
#include<stdio.h>
#include<string.h>
int axn[][6][2]={

        {{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},

        {{-1,-1},{100,6},{-1,-1},{-1,-1},{-1,-1},{102,102}},

        {{-1,-1},{101,2},{100,7},{-1,-1},{101,2},{101,2}},

        {{-1,-1},{101,4},{101,4},{-1,-1},{101,4},{101,4}},

        {{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},

        {{-1,-1},{101,6},{101,6},{-1,-1},{101,6},{101,6}},

        {{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},

        {{100,5},{-1,-1},{-1,-1},{100,4},{-1,-1},{-1,-1}},

        {{-1,-1},{100,6},{-1,-1},{-1,-1},{100,1},{-1,-1}},

        {{-1,-1},{101,1},{100,7},{-1,-1},{101,1},{101,1}},

        {{-1,-1},{101,3},{101,3},{-1,-1},{101,3},{101,3}},

        {{-1,-1},{101,5},{101,5},{-1,-1},{101,5},{101,5}}

        };//Axn Table

int gotot[12][3]={1,2,3,-1,-1,-1,-1,-1,-1,-1,-1,-1,8,2,3,-1,-1,-1,

   -1,9,3,-1,-1,10,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};  //GoTo table

int a[10];

char b[10];
```

```
int top=-1,btop=-1,i;
void push(int k)
{
  if(top<9)
    a[++top]=k;
}
void pushb(char k)
{
  if(btop<9)
    b[++btop]=k;
}

char TOS()

{

   return a[top];

}
void pop()
{
   if(top>=0)
     top--;
}
void popb()
{
   if(btop>=0)
     b[btop--]='\0';
}
void display()
{
  for(i=0;i<=top;i++)
    printf("%d%c",a[i],b[i]);
}
void display1(char p[],int m) //Displays The Present Input String
{
  int l;
  printf("\t\t");
  for(l=m;p[l]!='\0';l++)

    printf("%c",p[l]);
  printf("\n");
}
void error()
{
```

```
  printf("Syntax Error");
}
void reduce(int p)
{
   int len,k,ad;
   char src,*dest;
   switch(p)
   {
 case 1:dest="E+T";
        src='E';
        break;
 case 2:dest="T";
        src='E';
        break;
 case 3:dest="T*F";
        src='T';
        break;
 case 4:dest="F";
        src='T';
        break;
 case 5:dest="(E)";
        src='F';

        break;
 case 6:dest="i";
        src='F';
        break;
 default:dest="\0";
  src='\0';
  break;
   }
   for(k=0;k<strlen(dest);k++)
   {
     pop();
     popb();
   }
   pushb(src);
   switch(src)
   {
 case 'E':ad=0;
   break;
 case 'T':ad=1;
   break;
 case 'F':ad=2;
   break;
 default: ad=-1;
```

```
   break;
   }
  push(gotot[TOS()][ad]);
}

int main()
{
   int j,st,ic;
   char ip[20]="\0",an;
  // clrscr();
   printf("Enter any String\n");
   scanf("%s",ip);
   push(0);
   display();
   printf("\t%s\n",ip);
   for(j=0;ip[j]!='\0';)
   {
 st=TOS();
 an=ip[j];
 if(an>='a'&&an<='z') ic=0;
 else if(an=='+') ic=1;
 else if(an=='*') ic=2;
 else if(an=='(') ic=3;
 else if(an==')') ic=4;
 else if(an=='$') ic=5;
 else {
    error();
    break;
 }
   if(axn[st][ic][0]==100)
    {
      pushb(an);
      push(axn[st][ic][1]);
      display();
      j++;
      display1(ip,j);
    }
   if(axn[st][ic][0]==101)
    {
      reduce(axn[st][ic][1]);
      display();
      display1(ip,j);
    }
   if(axn[st][ic][1]==102)
    {
   printf("Given String is accepted \n");
```

```
 //  getch();
      break;
    }
 /*  else
   {
   printf("Given String is rejected \n");
      break;
    }*/
  }
return 0;
}
```

Output:

```
C:\gcc program4.c
C:\a.exe

Enter any String
a+a*a$

0                   a+a*a$
0a5                 +a*a$
0F3                 +a*a$
0T2                 +a*a$
0E1                 +a*a$
0E1+6               a*a$
0E1+6a5             *a$
0E1+6F3             *a$
0E1+6T9                     *a$
0E1+6T9*7                   a$
0E1+6T9*7a5                 $
0E1+6T9*7F10                $
0E1+6T9                     $
0E1                         $

Given String is accepted
```

Program 5:

AIM: To Design LALR bottom up parser for the given language.

```
<parser.l>
%{
#include<stdio.h>
#include "y.tab.h"
%}
%%
[0-9]+ {yylval.dval=atof(yytext);
return DIGIT;
}
\n|. return yytext[0];
%%
<parser.y>
%{
/*This YACC specification file generates the LALR parser for the program
considered in experiment 4.*/
#include<stdio.h>
%}
%union
{
double dval;
}
%token <dval> DIGIT
%type <dval> expr
%type <dval> term
%type <dval> factor
%%
line: expr '\n' {
```

```
printf("%g\n",$1);
}
;
expr: expr '+' term {$$=$1 + $3 ;}
| term
;
term: term '*' factor {$$=$1 * $3 ;}
| factor
;
factor: '(' expr ')' {$$=$2 ;}
| DIGIT
;
%%
int main()
{
yyparse();
}
yyerror(char *s)
{
printf("%s",s);
}
```

OUTPUT:

```
C:\ bison –d p.y
C:\ flex p.l
C:\gcc flex.yy.c p.tab.c
C:\a.exe
 2+3
5
```